Octave and Matlab

James Emery

9/6/2009

Contents

1 Introduction

Matlab is a program for doing numerical mathematics. Originally it was developed by Cleve Moler when he was at the University of New Mexico

and associated with Sandia Laboratories. It was designed for students in a course in Linear Algebra. It was written with a government grant and so was originally a public domain program. Now it is a very expensive commercial program. Along with matlab comes several *Tool Boxes* such as the Signal Processing Toolbox, the Image Processing Toolbox, and so on.

Both Matlab and Octave are probably best used by editing a script file in an editor. Matlab has an editor, and if you click on an "m" file, the Matlab editor will open it. Then in Matlab one can type the name of the "m" file, to run it, so for example if the file name is test.m, you would type just "text" to run it. When entering commands interactively in Matlab, one must worry about redefining variables. To be safe sometimes one should clear all the variables in the workspace.

Octave is a free GNU program for doing numerical mathematics whose commands are mostly compatible with **Matlab**. So much of a Matlab book can be used to learn Octave.

2 Downloading Octave

To download GNU Octave, Google Octave. You will then select

```
http://www.gnuorg/software/octave
Then select download.
You will go to:
http://www.gnuorg/software/octave/download.html
Scroll down to "Windows" and "Octave Forge," which select.
You will go to
http://octave.sourceforge.net/
Select "Windows Installer" (or " "Octave.app for MacOS X" for macintosh)
Note the location of the saved file, which for Windows is called "Octave-3.0.0-setup.exe"
Click the file to run it. Take all defaults. It will be installed. To run it select
Octave in the program menu, then run it or browse the manual in the html or pdf versions (616 pages).
```

3 A Few Command Examples

```
format long, this displays 15 digits on output,
format short, this displays about 5 digits
pwd % prints the current working directory. (a comment is indicated by a percent sign)
cd c:\je\m % change the current directory
scriptname % loads an m-file script called scriptname into the workspace ( do not use the m extension)
testOctave1 % there is a script test program in the c:\je\octave directory
myfunction(a,b) % loads an m-file function, and executes it with parameters a and b
x = -10:0.1:10;
plot (x, sin (x));
lambda=eig(a)
```

```
[v,lambda]=eig(a)
diary % diary turns on a log of the commands issued in the session, and stores it in a file called diary
diary off
type scriptname % types a script called scriptname if it has been loaded into the workspace
x=[1,2,3,4,5,6,7,8,9]
y=exp(x)
cd c:\je\m
who % lists the variables that are currently defined
cd \je\octave % changes the working directory to c:\je\Octave
global x % make a variable global so that it can be accessed inside a function
hold on % add the next plot to the current axis
writetofile %  calls the script writetofile.m located in the working directory.
quit % ends the session
```

4 Using Octave as a Calculator

Octave and Matlab can be used like a graphing calculator for doing textbook like scientific problems. For example consider finding the current in an alternating current network by combining impedances using complex numbers. Use the argument function to find phase angles. Change the frequency by backing up with the up arrow. Calculate the angle between the voltage and current phasers to calculate power factor. Turn on diary to record your calculation.

5 Scripts

A script is a file (an m file, with extension .m) containing Octave commands. When the name of the file is typed at the Octave command line, Octave fetches the file and executes all of the commands in the file just as if they were typed at the Octave command line. For example, here is a script called "writetofile.m" that illustrates the writing of data to files in a couple of ways.

```
x = 0:.1:1;
y = [x; exp(x)]; % the ending semicolon suppresses output
y % this displays the value of the matrix y
fid = fopen('c:\txt\exp1.txt','w');
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid);
%
y=y';
```

```
fid = fopen('c:\txt\exp2.txt','w');
for i = 1:11
 fprintf(fid,'%6.2f %12.8f\n',y(i,1),y(i,2));
end;
fclose(fid);
```

This was actually a script written for Matlab. It works in Octave, as most basic Matlab commands do. Typing the m-file name, without the m extension, such as by typing "write2file," creates two files in directory c:/txt

Here is a listing of, "/c:/txt/exp1.txt," which is one of the files produced by the script:

```
  0.00   1.00000000
  0.10   1.10517092
  0.20   1.22140276
  0.30   1.34985881
  0.40   1.49182470
  0.50   1.64872127
  0.60   1.82211880
  0.70   2.01375271
  0.80   2.22554093
  0.90   2.45960311
  1.00   2.71828183
```

The file is written with the Unix end of line characters, namely linefeeds (lf). So to convert to the windows end of line characters "crlf," one may use a utility program, like say my **C** program "lf2nl.c".

6 Linear Equation Solution

Consider the equations

$$2x + 3y + 4z = 9$$

$$x - y - z = -1$$

$$5x + y - z = 5$$

We can solve these using

```
a=[2,3,4;1,-1,-1;5,1,-1]
b=[9;-1;5]
ai=inverse(a)
c=ai*b
```

Now we computed the inverse of matrix a and then multiplied. This is an expensive calculation, because a solution to a linear set of equations can be solved without computing the inverse and multiplying. So the more efficient solution method is available in Matlab or Octave by using the backwards division operator:

```
c = a\b
```

7 Fourier Transform Example

The Fourier transform of the function f is defined as (Goldberg, **The Fourier Transform**)

$$g(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt.$$

By the Fourier integral theorem

$$f(t) = \int_{-\infty}^{\infty} g(\omega) e^{i\omega t} d\omega.$$

Example. Let

$$f(t) = \left\{ \begin{array}{cc} 2e^{-3t} & t \geq 0 \\ 0 & t < 0 \end{array} \right\}$$

Then

$$g(\omega) = \frac{2}{3 + i\omega}$$

Also see foran.tex

Here is a script file to compute the fft in matlab or ocatve:

```
%Example page 229, Mastering Matlab 5, fft
N=128;
t=linspace(0,3,N);
f=2*exp(-3*t);
Ts = t(2) - t(1);
```

```
Ws = 2*pi/Ts;
F=fft(f);
Fp=F(1:N/2+1)*Ts;
W=Ws*(0:N/2)/N;
Fa=2./(3+j*W);
plot(W,abs(Fa),W,abs(Fp),'+')
xlabel('Frequency,Rad/s')
ylabel('|F(\omega)|')
title('Figure 21.1; Fourier Transform Approximation')
```

8 Reading Data From a File as a Matrix

A data file consisting of numbers, with say m records, each consisting of n numbers, can be read into Octave or Matlab with the script rdmat.m, which is listed here. The numbers on a row, may be separated with blanks, or with a comma. So if data is saved from Excel as a comma delimited list, then it may be read into Octave as a matrix. So for example suppose one has some xy data in a file called a.dat in the Octave working directory (Remember, one can type "pwd" to print your working directory, and can change the working directory using "cd"). To plot this data one might do this:

```
a=rdmat('a.dat')
x=a(:,1)
y=a(:,2)
plot(x,y)
```

Or perhaps the coefficients of the left side and right side of a linear equation are stored in a.dat, and we wish to read in the data and solve the linear equation.

```
m=rdmat('a.dat')
a=m(1:3,1:3)
b=m(1:3,4)
x=a\b % solution of a*x = b
```

Here is the script rdmat.m:

```
%read a matrix from a file
% Example usage: m=rdmat('c:\je\octave\test.txt')
function a=rdmat(fname)
 fid=fopen(fname,'r');
 s1=fgets(fid);
 k=1;
 while(!feof(fid))
  if k == 1
   s=s1;
```

```
  else
   s=[s;s1];
  end
  s1=fgets(fid);
  k=k+1;
 end
 a=str2num(s);
endfunction
```

9 Solving a Differential Equation Numerically

To solve a differential equation numerically, one must first write it as a system of first order equations, and specify the initial conditions. Here we call lsnode to solve such an equation.

```
%solve differential equation from octave manual,
%where fde is defined in file fde.m
x0 = [1; 2]; % set initial conditions
% select 200 equally spaced time points between 0 and 50
t = linspace(0, 50, 200);
x = lsode ("fde", x0, t);
plot (t, x)
```

The right hand side of the differential equation is defined in a function script called fde.m. Note that the name of the function and the file name must agree.

```
function xdot = fde (x, t)
 r = 0.25;
 k = 1.4;
 a = 1.5;
 b = 0.16;
 c = 0.9;
 d = 0.8;
 xdot(1) = r*x(1)*(1 - x(1)/k) - a*x(1)*x(2)/(1 + b*x(1));
 xdot(2) = c*a*x(1)*x(2)/(1 + b*x(1)) - d*x(2);
endfunction
```

10 Damped One-Dimensional Vibration

The differential equation for damped vibration is

$$m\ddot{x} + c\dot{x} + kx = F(t).$$

The mass is m, the viscous damping constant is c, the stiffness is k, and the applied force is $F(t)$. For free damped vibration the roots of the characteristic equation are

$$-\frac{c}{2m} \pm \sqrt{(c/2m)^2 - k/m}.$$

The natural undamped resonant angular frequency is

$$\omega_n = \sqrt{k/m}.$$

The critical damping constant is

$$c_c = 2m\omega_n.$$

The critical damping ratio is defined to be

$$\zeta = \frac{c}{c_c}.$$

We have

$$c = 2\zeta m\omega_n,$$

so that if we divide the damped vibration equation by the mass m, we can write it in the form

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = \frac{F(t)}{m}.$$

Let us suppose that the force is zero $\zeta < 1$ so that we have a case of decaying oscillation. The equation becomes

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0.$$

Solving this equation by substituting in $\exp(pt)$ to get the characteristic polynomial in p, we find that

$$x = x_0 \exp(-\zeta\omega_n t) sin(\omega_d t + \phi),$$

where the angular frequency of damped oscillation is

$$\omega_d = \omega_n \sqrt{1 - \zeta^2},$$

and the corresponding period is

$$\tau_d = \frac{2\pi}{\omega_d}.$$

The notation in this section follows *Thomson*: **Theory of Vibration With Applications** Let us consider the case where $x(0) = 0$, so that the phase factor ϕ is zero, we have

$$x = x_0 \exp(-\zeta \omega_n t) sin(\omega_d t).$$

Let us take the natural frequency to be

$$\omega_n = 2\pi,$$

so that the natural period is

$$T_n = 1.$$

Suppose $\zeta = 1/2$, so that ω_d is about 5.4414, and the damped period is about

$$T_d = \frac{2\pi}{\omega_d} = 1.1547 \text{sec.}$$

Also suppose $x_0 = 1.$. Here is an octave script called dampedh.m:

```
zeta=1/16;
omega_n=2*pi;
omega_d=omega_n*sqrt(1.-zeta^2);
n=200;
t1=0.;
t2=3.;
t=linspace(t1,t2,n);
for k= 1:n
 x(k)=exp(-zeta*omega_n*t(k))*sin(omega_d*t(k));
end
plot(t,x)
```

Now let us compute this solution numerically. We want to solve

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0.$$

Given the value of x and its first derivative $\dot{(x)}$ at 0, there is a unique solution. The numerical solution should agree closely with

$$x(t) = \exp(-\zeta\omega_n t)sin(\omega_d t).$$

To solve a second order equation numerically, we must convert it to a first order system. So define

$$u_1 = x,$$

$$u_2 = \frac{dx}{dt}.$$

So we get the system

$$\frac{u_1}{dt} = u_2,$$

$$\frac{u_2}{dt} = \frac{d^2x}{dt^2} = -(2\zeta\omega_n u_2 + \omega_n^2 u_1).$$

For the closed form solution computed above, which we want the numerical solution to match, we have

$$x(0) = 0,$$

and the derivative of x is

$$\frac{dx}{dt} = -\zeta\omega_n \exp(-\zeta\omega_n t)\sin(\omega_d t) + \exp(-\zeta\omega_n t)\omega_d \sin(\omega_d t).$$

So

$$\frac{dx(0)}{dt} = \omega_d.$$

So the required initial conditions for our system are

$$u_1(0) = 0,$$

and

$$u_2(0) = \omega_d.$$

A script, called dampedhn.m, for solving the equation numerically, is

```
% dampedhn.m solving a differential equation
% for damped harmonic vibration numerically.
% requires script fdedh.m for the left side of the equation
% See octave.tex
zeta=1/16;
omega_n=2*pi;
omega_d=omega_n*sqrt(1.-zeta^2);
n=200;
t1=0.;
t2=3.;
% set the initial conditions
u0 = [0; omega_d];
% select n equally spaced time points between t1 and t2
t=linspace(t1,t2,n);
u = lsode ("fdedh", u0, t);
```

Before we can run this script we have to provide a definition of the left side of our DE system in the function "fdedh," in an m file with the same name.

```
%fdedh.m leftside of a differential equation for damped harmonic motion
%called by dampedhn.m
function udot = fdedh (u, t)
 zeta=1/16;
 omega_n=2*pi;
 udot(1) = u(2);
 udot(2) = -(2*zeta*omega_n*u(2) + omega_n^2*u(1));
endfunction
```

To compare the values from the closed form solution and the numerical solution we may put them in a two column matrix and list it.

```
 x1=x'; %convert x to a column vector
 u1=u(:,1) % select the first column of u
 for k=1:200 % put x1 and u1 into a 2 column matrix
  m(k,1)=x1(k)
  m(k,2)=u1(k)
 end
 m % list m
```

11 Bibliography

[1]Fausett Laurene V., **Applied Numerical Analysis Using Matlab**, Prentice-Hall, 1999.
[2]Hanselman Duane, and Littlefield Bruce, **The Student Edition of Matlab Version 4 User's Guide**, The Math Works, 1995.
[3]Hanselman Duane, and Littlefield Bruce, **Mastering Matlab 5, A Comprehensive Tutorial and Reference**, Prentice-Hall, 1998.
[4]Eaton John W., Bateman David, Hauberg Soren, **GNU Octave: A high-level interactive language for numerical computations**, Edition 3 for Octave version 3.0.0, July 2007 (PDF document).